CHAPTER 15

SQL Server 2005 Technology Primer

Microsoft SQL Server 2005 is comprehensive enterprise class data management and business intelligence platform. It provides a huge number of features and capabilities to manage data, develop applications integrated with the data, and finally analyze that data. It is also a complex and multifaceted platform.

> **Note**
>
> *Primum non nocere.*
>
> Attributed to the Greek physician Galen (AD 129–200)

The Latin quotation in the note translates to "First, do no harm." An administrator must ensure that changing the SQL Server 2005 system does not harm the system or the underlying data. To ensure that, the database administrator (DBA) must first understand the SQL Server 2005 system before attempting to change it.

With that level of capabilities, integration, and features of SQL Server 2005 comes a degree of complexity. This can make administration a challenging task for the DBA. Before being able to administer the platform, the DBA must understand how the product is put together.

This chapter lays out the architecture, features, dependencies, and integration between the various components. It covers the features of the SQL Server 2005 platform at a 10,000-foot view, allowing you to see the breadth of the product and how it all connects. The chapter also covers the various editions of SQL Server 2005 and how they differ.

This information will help you get your arms around the product conceptually. This information also paves the way for the other chapters in the book. For each component discussed in this technology primer, another chapter in the book goes into the more detailed administration and management of that component.

What Is New with Service Pack 2?

The book is based on Microsoft SQL Server 2005 Enterprise Edition with Service Pack 2. At a high level, the service pack enhances a number of areas including

- Data mining add-ins for Microsoft Office 2007 to allow Excel 2007 and Visio 2007 to directly access Analysis Services.
- Reporting Services integration with Office SharePoint Server 2007 to enable publication of reports into SharePoint sites.
- Data compression to reduce the space requirements for data warehouses.
- New business intelligence capabilities.
- New interoperability capabilities that allow Report Services and Report Builder to support Oracle and Hyperion Essbase cubes.
- Improved performance of components for better scalability.
- Improved manageability within database maintenance plans, management reports, and a new Copy Database Wizard.

Service Pack 2 includes many specific enhancements to the Database Engine, Analysis Services, Integration Services, Replication, and Reporting Services components. The specific enhancements to each of the components are covered in the specific chapter on each component.

Service Pack 2 also includes fixes to more than 130 different bugs in the SQL Server 2005 platform, which is a massive leap forward in the stability of the product.

Features of SQL Server 2005

The release of SQL Server 2005 included a number of features and enhancements over the previous version of the product. These features of SQL Server 2005 can be organized into three main categories: data management, development, and business intelligence.

Data Management

Data is not of much use if it cannot be accessed. Users and applications need to be able to access the appropriate data when they need it. SQL Server 2005 has been developed and engineered with a number of features that target the data management. They include

- **Availability**—Features that support availability are database mirroring, failover clustering, and replication. In addition, SQL Server 2005 increases availability by minimizing downtime through online indexing, online restores, fast recovery that allows users to access the database while incomplete transactions are being rolled back, and a dedicated administrator connection to ensure that DBAs can always access the server.
- **Performance and Scalability**—Performance and scalability of the database are critical, and SQL Server 2005 has a number of features that deliver lightning fast performance. These features include instant read-only database snapshots, snapshot isolation for rapid access to almost real-time data, replication to manage concurrent access to the data, and support for 64-bit platforms for large parallel memory access and larger memory spaces.
- **Security**—Security in today's uncertain world is also very important. SQL Server 2005 supports native encryption of the data within the database, granular permissions, and the highly secure Kerberos authentication. It also supports data encryptions of the data within the database and tight integration with Public Key Infrastructure (PKI) to encrypt data across the network. SQL Server 2005 deploys securely, with many services and features turned off by default to reduce the attack surface.
- **Manageability**—SQL Server 2005 helps DBAs manage the database infrastructure through a variety of tools such as the SQL Server Management Studio, SQL Server Configuration Manager, and Maintenance Plans. The platform makes it easier to manage large databases through sophisticated indexes and partitions. SQL Server 2005 is ready to be managed by Microsoft Operations Manager (MOM) out of the box with the SQL Server 2005 Management Pack to provide proactive monitoring and alerting.
- **Interoperability**—Working with other products and technologies is another key feature of SQL Server 2005. This includes support for industry standards such as Extensible Markup Language (XML) and

integration with other platforms such as Oracle. And SQL Server 2005 supports and leverages Microsoft technologies such as the Microsoft .NET Framework, Microsoft Windows Server, Microsoft Visual Studio, and Microsoft Office. The platform even includes a component dedicated to integration, the Integration Services component.

All these features taken together make SQL Server 2005 an enterprise class database management platform.

Development

Development is a difficult enough endeavor without having the tools and platform get in the way. Microsoft SQL Server 2005 delivers many features to assist the developer and streamline the processing of development.

The development features in SQL Server 2005 are engineered and built into the product from the ground up rather than bolted on as an afterthought. This means that they actually make the developer's job easier and more productive.

These features include

- **Integration**—The deep integration with Microsoft Visual Studio provides developers with an integrated tool for developing and debugging applications.
- **Improved tools**—Developers can use the same tools while developing Transact-SQL (TSQL), XML, Multidimensional Expression (MDX), and/or XML for Analysis (XMLA) code.
- **Language support**—The SQL Server 2005 database engine hosts the common language runtime (CLR), so developers have flexibility in the language they choose to develop in. This includes TSQL, Microsoft Visual Basic 2005, and Microsoft Visual C#. Developers can even leverage non-Microsoft code for rapid application development.
- **XML and Web Services**—SQL Server 2005 supports XML natively with the new XML data type for end-to-end seamless integration of data with internal and external systems. It offers support for open standards such as HTTP, XML, Simple Object Access Protocol (SOAP), XQuery for querying the XML data type, and the XML Schema Definition (XSD) for communication across all systems.

These features increase developers' productivity and reduce the time-to-market for applications and services.

Business Intelligence

Business intelligence (BI) allows organizations to leverage the vast amounts of data they collect and gain insights from it through analysis. This includes pooling together the heterogeneous data repositories and providing user-customizable views of the data and insight into trends and patterns hidden in the data. Ultimately, this improves the agility, efficiency, and performance of the organization. This is a big promise.

Microsoft SQL Server 2005 delivers on this promise through a suite of features that target business intelligence. These features include

- **Integration**—With SQL Server 2005 Integration Services, data can be moved to and from a variety of sources such as OLE DB, ADO.NET, ODBC, flat files, Excel sheets, and XML. While moving, the data can be transformed with operations such as aggregating, sorting, conversion, merging, and auditing.
- **Analysis**—Through the SQL Server 2005 Analysis component, the platform delivers an integrated view of business data for reporting, online analytical processing (OLAP), key performance indicators (KPIs), and data mining. One of the key enablers of this is the Unified Dimensional Model (UDM), which provides a meta repository of all business entities, logic, calculations, and metrics.
- **Reporting**—A key requirement of business intelligence is to get information displayed in a meaningful and accessible manner. Microsoft SQL Reporting Services provides the facilities to create, manage, and deliver reports. These reports can be static reports or web-enabled interactive reports.
- **Data Mining**—Data mining allows the organization to explore the data. SQL Server 2005 delivers that capability via the data mining elements in SQL Analysis Services. This includes a data mining object with multiple mining techniques, such as clustering, linear regression, neural network, and naïve bayes. It also includes a Data Mining Wizard to generate analysis such as market basket analysis, churn analysis, forecasting, and data quality analysis.
- **Data Warehousing**—Data warehouses have unique concerns due to their size and complexity. SQL Server 2005 has a number of features to support data warehousing, including table partitions for fast data loading, online indexing, improvements in the TSQL to support new data types and analytic functions, and fine-grained backup and restore operations.

These improvements allow SQL Server 2005 to deliver the features, tools, and functionality to deliver business intelligence to the enterprise.

64-Bit Computing

SQL Server 2005 supports both 32-bit and 64-bit platforms, running natively on both. The 64-bit hardware and 64-bit Windows Server 2003 operating system are the de facto standard in high-performance computing. SQL Server 2005 is optimized to run on the 64-bit platform and take advantage of all the architectural advantages it offers.

The advantages of the 64-bit version of SQL Server 2005 over the 32-bit version of SQL Server 2005 are

- A larger, directly addressable memory space. The 32-bit version supports up to 4GB of memory, whereas the 64-bit version can support up to 1024GB of memory. This allows for better performance running complex queries and database operations.
- A higher degree of parallelism, allowing the 64-bit version to support up to 64 processors and better performance per processor than 32-bit systems.
- The wider 64-bit bus architecture allows more data between the cache and the processors in less time.

With all these advantages, the SQL Server 2005 64-bit platform can handle the largest enterprise and scientific workloads, support database consolidation, and scale to meet the most demanding database applications.

Components in SQL Server 2005

The features and functionality of the SQL Server 2005 database platform are divided into these components:

- Database Engine
- Analysis Services
- Reporting Services
- Notification Services
- Integration Services
- Full-Text Search
- Replication
- Service Broker

The components and their integration are shown in Figure 15.1. These components help organize and structure the vast array of tools and technology the platform provides to DBAs, developers, and business analysts. The following sections describe each of these components and key structures in more detail.

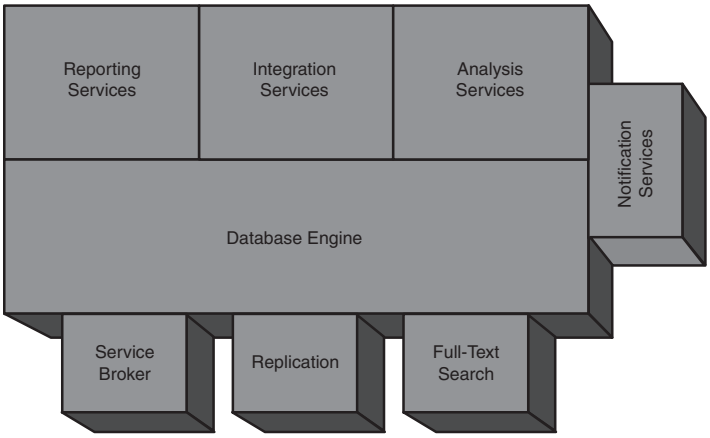

FIGURE 15.1
SQL Server 2005 components.

Database Engine Component

The Database Engine component, shown in Figure 15.2, is the heart and soul of the SQL Server 2005 product. It is the core service for storing, processing, and securing data. It is designed to provide a scalable, fast, and highly available platform for data access and the other components.

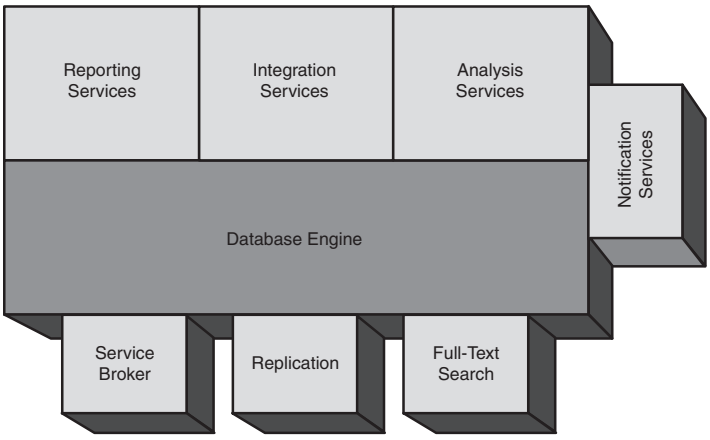

FIGURE 15.2
Database Engine component.

The following sections cover some of the technology and features of the Database Engine component.

Databases

Databases manage, process, and store the data that is the rationale for the entire SQL Server 2005 database platform. The data is stored and organized with a variety of structures to facilitate the access, performance, and recoverability of that data. These structures include

- **Tables**—Databases in SQL Server 2005 are composed of tables. Tables consist of data in rows and columns. Rows are also called *records* and columns are also called *attributes*. Each column stores particular types of information in the table, such as a name, date, or value. A database can have many tables.
- **Indexes**—Indexes allow data to be rapidly located in a table or view. The indexes are built from one or more columns in a table or view. They speed up query performance of the database and can also ensure that the values in a particular column are unique.
- **Partitioned Tables and Indexes**—Tables and indexes can be partitioned into horizontal units that are groups of rows. This makes it easier and faster to use large sets of data. Partitions are frequently organized by date, such as separate partitions for monthly or daily data.
- **Views**—A view is a virtual table created by a query statement. It is similar to a table but does not contain any data unless it is indexed. It allows different combinations of rows and columns to be created without having to replicate the data in the tables. However, views can be manipulated just like tables, and the underlying table data will be modified. Views can be standard, indexed, or partitioned.
- **Stored Procedures**—Stored procedures are code that is stored within the database and can be executed remotely. They improve the security of the database because they allow the code to be centrally reviewed and managed. Stored procedures also protect the application from SQL injection–type attacks. They run locally on the server and can reduce network traffic. They can be written as either TSQL or CLR types.
- **Triggers**—Triggers execute code in response to certain events, specifically logon events, data definition language (DDL) events, and data manipulation language (DML) events. Collectively, the triggers allow DBAs and developers to respond to events in the database such as logons and changes to the database. They are useful for auditing and regulating database operations.

- **Event Notifications**—Similar to triggers, event notifications respond to DDL and SQL trace events. However, event notifications do not execute code but rather send messages to the Service Broker. They also execute outside the scope of the transactions that trigger them, allowing them to execute out of band and even remotely.
- **Assemblies**—These objects reference managed application modules (dynamic linked library, or DLL, files) that are created in the .NET Framework CLR. They are used to execute CLR-managed code.

Databases are stored in data and log files. The data files contain data and related objects such as tables, indexes, and views. The log files contain the transaction logs that allow the database to be recovered in the event of a problem. Each database can have multiple data and log files, but there must be at least one of each.

A database is always in a specific state, such as ONLINE. These states indicate if the database is available and the primary cause for the lack of availability. The states are shown in Table 15.1.

Table 15.1 **Database States**

State	Definition
ONLINE	Database is available for access.
OFFLINE	Database is unavailable. The database is taken offline by the DBA.
RESTORING	The database is being restored and is unavailable.
RECOVERING	The database is being automatically recovered and is unavailable. If the recovery fails, then the database will become SUSPECT.
RECOVERY PENDING	The recovery process is pending user action, and the database is unavailable.
SUSPECT	The database failed the automatic recovery and is unavailable.
EMERGENCY	The DBA has set the status of the database to emergency, and the database is unavailable. This is typically used for troubleshooting.

System Databases

Some databases are more special than others, specifically the system databases. These databases are used to manage the SQL Server 2005 database engine and databases.

These databases can be found in the System Databases folder in the SQL Server Management Studio. Table 15.2 lists the systems databases and their functions.

Table 15.2 **System Databases**

System Database	Description
master	This database records all the system-level information for an instance of SQL Server.
msdb	This database is used by SQL Server Agent for scheduling alerts and jobs.
model	This database is used as the template for all databases created on the instance of SQL Server. Modifications made to the model database, such as database size, collation, recovery model, and other database options, are applied to any databases created afterward.
resource	This read-only database contains system objects that are included with SQL Server 2005. System objects are physically persisted in the Resource database, but they logically appear in the sys schema of every database. This database is not visible in the Microsoft SQL Server Management Studio System Databases folder.
tempdb	This database is used as a workspace for holding temporary objects or intermediate result sets.

Note

SQL Server does not support users directly updating the information in system objects such as system tables, system stored procedures, and catalog views.

Federated Database Servers

The federated database functionality allows DBAs to spread the processing load of a highly available application across a group of servers. The data is horizontally spread across the federation member servers. However, the federated database servers present to the application a single cohesive view of the database.

Actually achieving the high performance possible with federated database servers requires that the application send SQL statements to the server that

holds the data required by the statement. This requires careful design and partitioning of the database and views, as well as design of the application.

Security

Microsoft SQL Server 2005 has been designed and implemented with security in mind. The security ranges from granular assignment of permissions to encryption of data within the database to enforcement of password policies. The combination of these security features makes SQL Server 2005 one of the most secure database platforms available.

The database engine uses a variety of elements to control security. These security elements are

- **Principals**—Principals are the subjects that can request SQL resources. This can include Windows users and groups, SQL Server logins, and database principals. These can be an individual (such as a Windows user) or collections (such as a Windows group). Each principal is represented by a security identifier (SID).
- **Securables**—Securables are the objects (SQL resources) that must be protected and to which the SQL Server Database Engine component regulates access. The securables are organized into scopes, which include server, database, and schema. Server scopes can contain logins and databases. The database scope includes securables such as user, role, assembly, and a host of others. The schema scope includes the securable's type, XML schema collection, and object. The object securable members include constraint, function, queue, and others.
- **Permissions**—SQL Server 2005 has a detailed permissions structure to apply fine-grained control of access to securables. The permissions available to SQL are shown in Table 15.3. These permissions are applied to base securables and container securables to easily scope the assignment of permissions.

Table 15.3 **Permissions**

Permission	Definition
ALTER ANY	Grants ability to CREATE, ALTER, or DROP to individual securables in a database or a server.
ALTER	Grants the ability to alter properties of a securable, except for the ownership. This includes the ability to CREATE, ALTER, or DROP any securables or objects contained within the scope.

Table 15.3 **continued**

Permission	Definition
BACKUP/DUMP	Grants the ability to back up the securable.
CONTROL	Grants ownership-equivalent capabilities to the securable.
CREATE	Grants permission to create securables.
DELETE	Grants permission to delete securables.
EXECUTE	Grants permission to execute the securable.
IMPERSONATE	Grants permission to impersonate the login or user.
INSERT	Grants permission to insert data into the securable.
RECEIVE	Grants permission to receive Service Broker messages.
REFERENCE	Grants permission to reference the securable.
RESTORE/LOAD	Grants permission to restore the securable.
SELECT	Grants permission to view data within the securable.
TAKE OWNERSHIP	Grants permission to take ownership of the securable.
UPDATE	Grants permission to change data within the securable.
VIEW DEFINITION	Grants permission to view the securable definition.

- **Permission Hierarchy**—The SQL Server 2005 permissions are set at the level of servers and databases but can be set on a much finer-grained basis on all the securables. This allows the security of servers and databases to be locked down to the degree needed by even the most security-demanding application. The hierarchy allows the permissions to be assigned and flow down the hierarchy to reduce the administrative overhead and maintenance.
- **Fixed Database Roles Permissions**—For ease of administration, there are some predefined database-level roles in SQL Server 2005. These roles allow standard sets of permissions to be assigned easily. The fixed database roles are assigned permissions as given in Table 15.4.

Table 15.4 **Fixed Database Role Permissions**

Fixed Database Role	Database Permissions	Server Permissions
db_accessadmin	ALTER ANY USER, CREATE SCHEMA	VIEW ANY DATABASE
db_accessadmin	CONNECT	

Fixed Database Role	Database Permissions	Server Permissions
db_backupoperator	BACKUP DATABASE, BACKUP LOG, CHECKPOINT	VIEW ANY DATABASE
db_datareader	SELECT	VIEW ANY DATABASE
db_datawriter	DELETE, INSERT, UPDATE	VIEW ANY DATABASE
db_ddladmin	ALTER ANY ASSEMBLY, ALTER ANY ASYMMETRIC KEY, ALTER ANY CERTIFICATE, ALTER ANY CONTRACT, ALTER ANY DATABASE DDL TRIGGER, ALTER ANY DATABASE EVENT, NOTIFICATION, ALTER ANY DATASPACE, ALTER ANY FULLTEXT CATALOG, ALTER ANY MESSAGE TYPE, ALTER ANY REMOTE SERVICE BINDING, ALTER ANY ROUTE, ALTER ANY SCHEMA, ALTER ANY SERVICE, ALTER ANY SYMMETRIC KEY, CHECKPOINT, CREATE AGGREGATE, CREATE DEFAULT, CREATE FUNCTION, CREATE PROCEDURE, CREATE QUEUE, CREATE RULE, CREATE SYNONYM, CREATE TABLE, CREATE TYPE, CREATE VIEW, CREATE XML SCHEMA COLLECTION, REFERENCES	VIEW ANY DATABASE
db_denydatareader	Denied SELECT	VIEW ANY DATABASE
db_denydatawriter	Denied DELETE, INSERT, UPDATE	
db_owner	CONTROL	VIEW ANY DATABASE
db_securityadmin	ALTER ANY APPLICATION ROLE, ALTER ANY ROLE, CREATE SCHEMA, VIEW DEFINITION	VIEW ANY DATABASE

- **Fixed Server Role Permissions**—Similar to the fixed database roles, there are fixed server roles to assign standard sets of permissions. These server roles and their permissions are listed in Table 15.5.

Table 15.5 **Fixed Server Role Permissions**

Fixed Server Role	Server Permission
bulkadmin	ADMINISTER BULK OPERATIONS
dbcreator	CREATE DATABASE
diskadmin	Granted: ALTER RESOURCES
processadmin	ALTER ANY CONNECTION, ALTER SERVER STATE
securityadmin	ALTER ANY LOGIN
serveradmin	ALTER ANY ENDPOINT, ALTER RESOURCES, ALTER SERVER STATE, ALTER SETTINGS, SHUTDOWN, VIEW SERVER STATE
setupadmin	ALTER ANY LINKED SERVER
sysadmin	CONTROL SERVER

- **Encryption Hierarchy**—To protect the data, SQL Server 2005 supports a hierarchical encryption model. This encryption model starts at the Windows operating system level and SQL server level and then encrypts the data at the database level using a combination of certificates, asymmetric keys, and symmetric keys.
- **Context Switching**—The context of a user determines his or her access rights. SQL Server 2005 supports the ability to switch contexts by using the `EXECUTE AS` statement. This allows users to run in a lower privilege context for normal activities to reduce the security risk and then elevate to a higher privilege context when needed (with the appropriate credentials, of course) for specific tasks.
- **Module Signing**—This security feature allows modules to be signed within a database. This allows the modules' authenticity to be verified prior to execution to ensure that they have not been altered or corrupted.
- **Password Policy**—SQL Server 2005 can use the Windows operating system password policy mechanisms, such as password complexity and password expiration. This allows administrators to use the `ALTER LOGIN` statement to configure password policy on a login. This ensures those SQL passwords are secure and not easily compromised.

The net result of the security features is that SQL Server 2005 is highly secure. SQL Server has been evaluated against the Common Criteria Evaluation Assurance Level 1 (EAL1) and Level 4 (EAL4) and can be configured to comply with the Federal Information Processing Standard 140-2 (FIPS 140-2).

Analysis Services Component

The SQL Server 2005 Analysis Services (SSAS) component provides online analytical processing (OLAP) and data mining. OLAP is a modification of the original database concept of online transaction processing (OLTP). OLAP is designed to provide immediate answers to analytical and ad hoc queries from a multidimensional cube known as an *OLAP cube*. Data mining is the process of searching large volumes of data for patterns and trends. SSAS allows SQL Server 2005 to provide both these capabilities and is the core feature of business intelligence.

The Analysis Services component interfaces with the database engine component, the Integration Services component, and the Notification Services component, as shown in Figure 15.3.

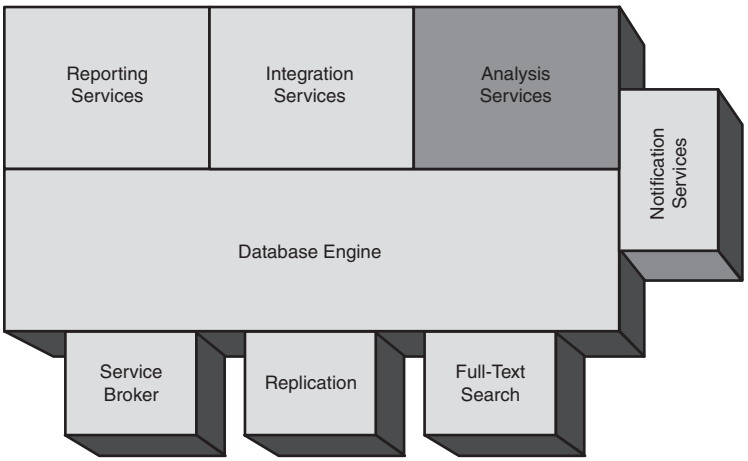

FIGURE 15.3
Analysis Services component.

The Analysis Services component gives a consistent view of data that supports metrics, data mining, and analytics. The elements that deliver these capabilities are discussed in the following sections.

Unified Dimensional Model

The Unified Dimensional Model (UDM) allows you to see a common view of an organization's data, one that hides all the underlying details and forces them to conform to business logic.

The UDM allows you to step back from having to understand where the data is specifically located, how it is stored, and the constraints on the values of the data. The UDM allows not only view access of the data, but also allows write access to the underlying data sources if appropriate.

It allows data to be organized via

- **Hierarchies**—Application of hierarchies to the data to allow drill-down into the data for more details.
- **Categories**—Categorization of data from disparate sources and formats into recognizable groupings such as employee data or customer data.
- **Natural time**—Natural time calendars, such as by quarter or fiscal year.
- **Language**—Automatic translation for the underlying data to support international data and access to the data so that, for example, a French user would see attributes in French and a U.S. user would see attributes in English.
- **Perspectives**—Division of the UDM into perspectives, allowing the UDM to show only the data and views relevant to the target users.
- **Key performance indicators**—Establishment of key performance indicators (KPIs) for data in the model, such as a 25% profit margin on sales. The model would automatically display the status of the metric as compared to the KPI.
- **Caches**—Improvement in performance by caching the data as the user browses the model.
- **Access control**—Control of access based on roles and permissions defined within the model, such as preventing users in competing subsidiaries from viewing each other's data.

The UDM can be built over data sources within different SQL Server 2005 servers, but also across other database servers such as Oracle or DB2.

The UDM simplifies and standardizes the access to data for users, providing an abstraction layer with lots of controls and features.

Server and Client Communications

The SSAS server runs as a service and consists of the following subcomponents:

- Security components for managing security
- XMLA Listener, which is an XML for Analysis listener
- Web Services Middle Tier for scalability and support for a broad set of languages
- Query Processor for processing queries

The server processes transactions, manages the metadata, performs intermediate computations, aggregates data, schedules queries, and caches objects.

The SSAS client is a thin client, with all the analysis computations taking place on the server. There is only a single roundtrip between the client and server, which allows service to scale without affecting client performance even as query complexity increases.

The SSAS client communicates with the SSAS server via XMLA. This can be XMLA over TCP/IP or XMLA over HTTP via an IIS server.

Applications can use any programming language to communicate with SSAS that uses one of the interfaces listed in Table 15.6 by using the SSAS Web Architecture middle tier. These providers utilize IIS to support the translation into XMLA for the SSAS.

Table 15.6 **Application Language Support and Interfaces**

Language	Interface
C++	OLE DB for OLAP
Visual Basic 6	ADO MD
.NET languages	ADO MD.Net
SOAP (any language)	XMLA

Regardless of the interface or client, all communication with SSAS is ultimately conducted in XMLA.

Data Sources

An SSAS data source is a connection to a physical data source. Connections can be made via a managed Microsoft .NET Framework provider or via a native OLE DB provider.

Data sources can include

- Microsoft SQL Server 2005 databases
- Microsoft SQL Server 2000 databases

- Oracle databases
- DB2 databases
- Teradata databases
- File sources

Data sources allow SSAS to analyze and aggregate data from a wide variety of physical sources into an integrated and cohesive view.

Data Source Views

A data source view is a virtual schema used by SSAS for the Analysis Services database objects such as cubes, dimensions, and mining structures (discussed in the following sections). These are used within the Unified Dimensional Model and by other components to present a virtual view of the underlying physical data from disparate sources.

Data source views consist of the following:

- Metadata representing objects from the physical data sources
- Metadata from multiple physical sources integrated into a single logical view
- Calculated columns, primary keys, and queries that are not in the physical data sources

In addition, data source views cannot be seen directly by client applications.

The data source views can be used by SQL Server 2005 Analysis Services, Integration Services, and Reporting Services.

Cubes, Dimensions, and Measures

Cubes are composed of dimensions and measures. The dimensions divide the cube, shown in Figure 15.4, into different regions. The measures are the specific values in the cell of the cube.

The cube structure allows the data to be organized in multiple ways (dimensions) at once, allowing for quick aggregation and analysis of the data.

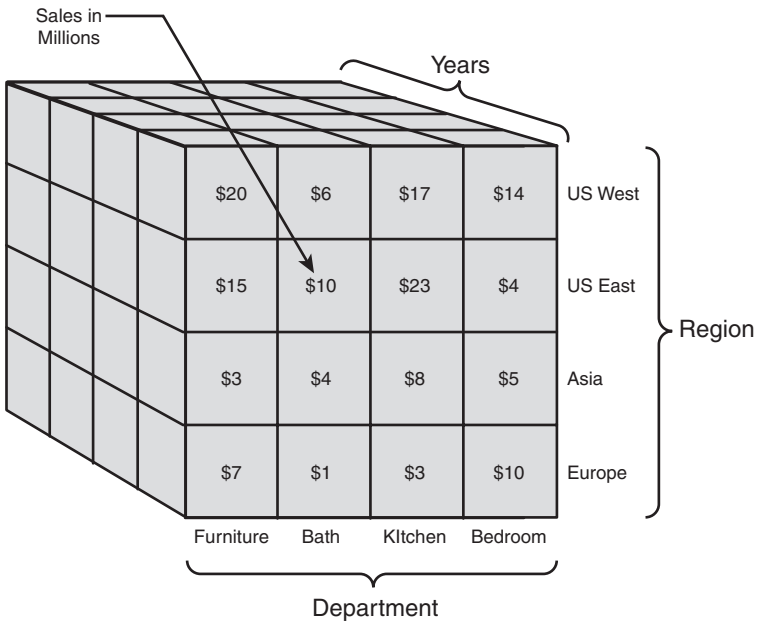

FIGURE 15.4
OLAP cube.

Mining Structures and Models

The objects used to do data mining in SQL Server 2005 include mining structures and mining models. Data mining structures are different from cubes. The data mining structure is composed of the mining structure columns, nested tables, and data mining models. The mining structure is used to organize the data and indicate where the source data is located, as well as how the data is used in the mining models.

The data mining model uses a mining model algorithm to actually mine the data represented in the mining structure. There can be more than one mining model in a given mining structure. The mining algorithms do the following:

- Classification algorithms predict one or more discrete variables.
- Regression algorithms predict one or more continuous variables, such as profit or loss.
- Segmentation algorithms organize the data into groups of items that have similar properties.

- Association algorithms find correlations between different attributes in the data.
- Sequence analysis algorithms summarize frequent sequences in the data.

The data mining objects can be created and manipulated in the data mining designer tool in the Business Intelligence Development Studio.

Roles

To simplify security, SQL Server 2005 Analysis Services uses role-based access control. The roles are used to link Windows users and groups to specific sets of rights and permissions to objects in Analysis Services. There are server roles and database roles.

The server role gives members full administrative rights over all objects in Analysis Services. This is used for administration, maintenance, and security activities on the particular instance of Analysis Services. The server role is predefined for each instance of Analysis Services and cannot be deleted, and the permissions cannot be changed.

The database roles are much more flexible than the server roles. The database roles are created as separate objects in the Analysis Services database. Users and groups can be added to the roles, and the permissions can be defined to permit the appropriate level of access for each role.

Reporting Services Component

The Microsoft SQL Server 2005 Reporting Services (SSRS) component, shown in Figure 15.5, allows for the presentation and delivery of data in a variety of ways. The reports can include tables, matrices, and free-form data. The reports' source data can include the Database Engine component, the Analysis Services component, or any Microsoft .NET data provider such as ODBC or OLE DB to access data sources such as Oracle or file-based data.

Output formats for the reports include

- HTML
- Excel
- PDF
- TIFF
- CSV

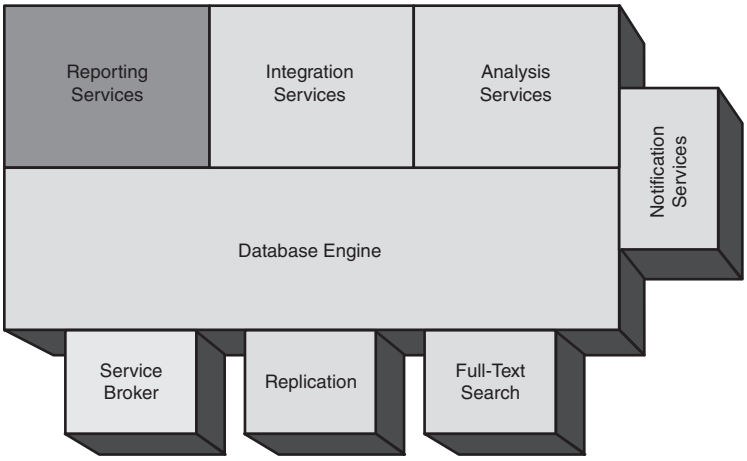

FIGURE 15.5
Reporting Services component.

The reports can be created, viewed, published, and managed through a web interface, the Report Manager. The SSRS component can also be controlled programmatically via the SSRS API.

The features and functions of the SQL Server 2005 Reporting Services can be modified through extensions. There are a variety of extension types, including security, data processing, rendering, report processing, and delivery.

The components of the Reporting Services are covered at a high level in the following sections. For a more detailed discussion of the SSRS, refer to Chapter 3, "Administering SQL Server 2005 Reporting Services."

Report Server

The report server is composed of two separate services: a web service and a Windows service. The web service facilitates user and programmatic access to the report server. The Windows service runs the scheduled reports, delivers reports, and runs maintenance tasks.

The report server has three separate classes of subcomponents, including processors, data storage, and extensions. This server can be tailored to specific needs by creating custom extensions. This allows the custom rendering to a specific format or delivery of reports to a new destination type.

Processors

Processors are the workhorses of the report server. There are two processors. When a report is requested, the report processor gets the report definition, retrieves the data specified in the report, formats the data as specified in the report definition, and finally renders the report into the specified format. The scheduling and delivery processor runs the scheduled reports and handles the delivery of the reports.

Data Storage

The reporting server database (ReportServer) contains all the properties, objects, and metadata for the reports. This includes published reports, report models for self-service generation of reports, and the report folder hierarchy. Multiple report servers can use the same database to ensure a consistent view of reports in a scaled-out environment.

Extensions

Extensions allow DBAs to extend or alter the functionality of the SQL reporting server. The report server must have at least one of each of the extension types, which include the security, data processing, rendering, delivery, and report processing extensions.

The security extension determines the method by which SQL Reporting Services authenticates and authorizes users. The default is Windows authentication. There can be only one security extension for each reporting service.

Data processing extensions each handle the query to different data sources such as SQL Server and ODBC. The data processing extensions not only retrieve the data, but also process it to transform it into a standard format for the report processor to use. The data processing extensions, no matter what data source the query, return a flattened row set for the report processor to use.

The rendering extensions convert the report data to a specific format such as HTML, Excel, CSV, XML, Image, or PDF. The image rendering extension defaults to TIFF but can output BMP, EMF, GIF, JPEG, PNG, or WMF as well.

Delivery extensions are used to deliver finished reports via email or saved to a file share. The email extension uses SMTP to deliver the report. The file share delivery extension saves the report to a network file share.

There are also report processing extensions that can be used to extend the standard tables, charts, lists, and other report types. For example, a custom

report processing extension could be used to create a map report using Microsoft MapPoint.

Notification Services Component

Notification Services generates notifications about events. This allows SQL Server 2005 users and administrators to define what they're interested in being notified about and then having SQL Server 2005 (through the Notification Services) inform them when those events occur.

The architecture of Notification Services, shown in Figure 15.6, is composed of four main elements: subscriptions, event collection, processing of the subscriptions, and finally delivery and formatting of the notifications.

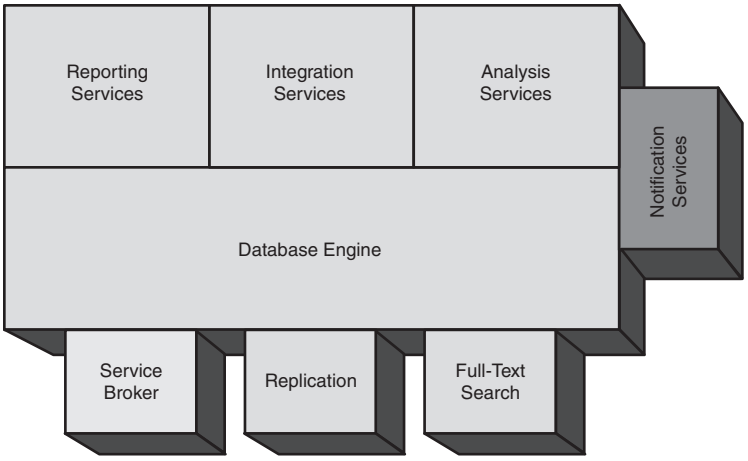

FIGURE 15.6
Notification Services component.

These elements allow notifications to be handled in a sophisticated and scalable way in SQL Server 2005. To be clear, Notification Services is more of a programming framework than a user interface. Development work needs to be done to leverage its features.

Subscriptions

The subscriptions are managed though subscription management interfaces. This is typically an ASP.NET web application that uses the Notification

Services subscription management API. This API exposes the subscription management objects such as

- **Subscriber**—An object that represents a subscriber. This can represent a single user, a process, or a group of users.
- **Subscriber Device**—This is a notification target, such as an email address or pager.
- **Subscription**—This is the definition of the event that the subscriber wants to be notified on, when to be notified, and by which device to be notified.

The interfaces can be designed as self-service applications, allowing subscribers to sign themselves up to notifications.

Event Collection

The event collection mechanisms allow information from databases, XML data load information from the XML API, or application events to generate notification events. The definitions of what to collect are called *event providers*.

Three standard event providers come with Notification Services, allowing notification based on database information, XML file data, and cube data. These standard event providers are

- **File System Watcher Event Provider**—This provider monitors a defined directory for the addition of an XML file, reads the file, and creates events based on the file contents.
- **SQL Server Event Provider**—This provider runs a defined TSQL query against a database and triggers events based on the results of the query.
- **Analysis Services Event Provider**—This provider uses an MDX query against an Analysis Services cube and triggers events based on the results of the query.

If the standard event providers are not sufficient, custom event providers can be developed.

When the event providers collect events, they place them in the event table.

Subscription Processing

With the events placed in the event table by the event providers, the subscription processing can be done by the generator. The generator evaluates the

events against the subscriptions to determine if notifications need to be generated.

The generator writes the notifications to a notification table. Optionally, the generator can also copy the events from the event table to a chronicle table to store the history.

Notification Formatting and Delivery

Finally, the notification is ready to be delivered to the subscriber. The formatting and the distribution of the notifications are handled by the distributor. The distributor supports a number of delivery channels and protocols, which are

- **SMTP Protocol**—This delivery protocol writes the notification to an SMTP service.
- **File Protocol**—This delivery protocol writes the notification to a file.
- **HTTP Protocol**—This delivery protocol is used to develop custom delivery channels such as SOAP, .NET Alerts, and SMS.

Before distributing the notification, the distributor formats the message. The formatted message can contain data from the event, computed data, or data generated by the content formatter. The distributor uses Extensible Stylesheet Language Transformation (XSLT) forms to generate the formatted messages.

Integration Services Component

The SQL Server 2005 Integration Services (SSIS) component, shown in Figure 15.7, integrates data from different sources. This integration includes importing, exporting, and transforming data from disparate sources. The data can be copied, merged, restructured, and cleaned as part of the integration processing, which makes the integration services a powerful tool in the development of data warehouses.

An important mention is that the Integration Services component fills an important gap in the extract, transform, and load (ETL) tool area from previous versions of SQL Server.

SSIS includes a number of capabilities and structures to make it easier to transfer and manipulate data that is being integrated.

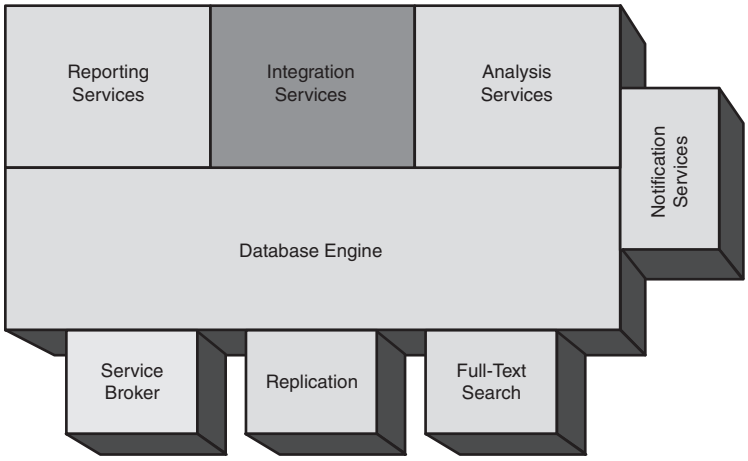

FIGURE 15.7
Integration Services component.

Integration Services Object Model

The SSIS provides a rich object model for developing integration jobs or packages. This allows DBAs to create sophisticated data integrations that are needed to merge complicated data sources.

The objects that make up the SSIS ecosystem are

- **Packages**—SSIS uses a package model, with the integration jobs being stored and run as packages. The package is the overarching organizational object and all the other objects are contained within a package.
- **Control and Data Flow**—Packages contain control flow and data flow tasks, which basically take data from a source and transform it to a destination. A multitude of different control and data flow tasks listed in the toolbox are ready for immediate use in the package.
- **Connection Managers**—The sources and destinations are affected using connection managers, which specify the connection to different types and sources of data.
- **Variables**—Variables within the packages can be used at runtime to store data and control scripts such as `For` loops.
- **Event Handlers**—There are a number of predefined event handlers, such as `OnError`, `OnWarning`, and `OnTaskFailed`. These event handlers are used to control package behavior programmatically within the

package. They allow the package to fail gracefully or monitor progress of the package run.

- **Log Providers**—Packages and even containers within a package can be logged for tracking and auditing. The logs can be written to text files, the Windows Event log, SQL Server, an XML file, or even the SQL Server Profiler.

Packages are designed using the SSIS Designer graphical tool in the Business Intelligence Development Studio.

Integration Services Service

The Integration Services (IS) service is a Windows service that manages the packages. It facilitates the following functions:

- Starting, monitoring, and stopping packages
- Importing and exporting packages
- Organizing packages into folders

The IS service is accessed using the SQL Server Management Studio tool. The various packages and their status can be viewed and organized there.

Full-Text Search Component

The Full-Text Search component allows fast searches against unstructured text data with SQL Server 2005 databases. This allows you to search for information using natural language. The fast-text search component actually produces results faster than `LIKE` searches of unstructured data through the optimizations in the indexing process.

The Full-Text Search component integrates with the Database Engine component, as shown in Figure 15.8.

The capability of Full-Text Search requires several different processes to gather the information, process the data, and facilitate the searches.

Full-Text Engine Process

The full-text engine populates and manages the full-text catalogs. The full-text engine also makes the full-text searches easier by maintaining indexes, a thesaurus, noise words, and linguistic analysis of the full-text indexes.

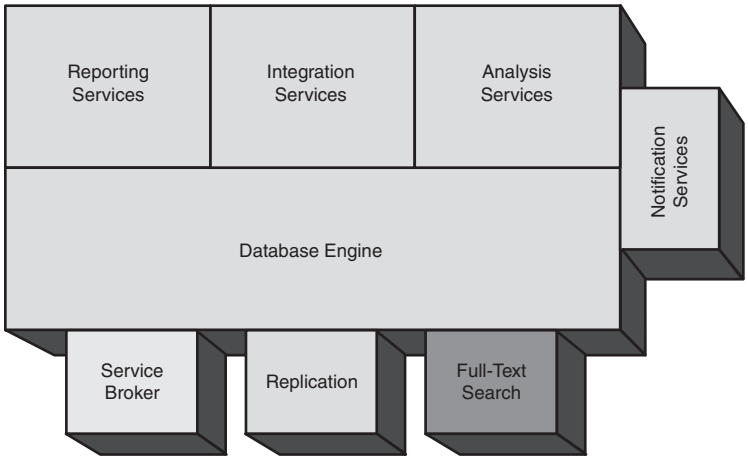

FIGURE 15.8
Full-Text Search component.

The thesaurus consists of an XML file of search term synonyms, one for each language supported. These files are mostly blank, except for synonymous Internet Explorer terms. They need to be populated manually to be used by the full-text search engine.

The noise words, which are also maintained in an XML file, basically are words that don't help the search, so they are left out of the indexes. The list of words is used to reduce the size of the indexes.

The linguistic analysis consists of word breaking and stemming to allow effective searches. The word breaking finds word boundaries that are appropriate for the language in use. Stemming conjugates verbs properly to ensure that the search terms are found in their various forms.

The full-text engine process populates the full-text indexes based on requests made directly by the SQL database engine. This process of indexing the data to create the full-text indexes is also known as *crawling*.

The full-text engine process also manages the filter daemon process, described next.

Filter Daemon Process

The filter daemon process is used during the full-text index generation process to access and filter the source data. The protocol handler portion of

the daemon accesses the databases that are being indexed. The filters portion of the daemon extracts the actual information from the data, eliminating the nontextual and formatting information.

SQL Server Process

The SQL Server process handles client requests to search the full-text indexes. The query is passed off to the full-text engine process and the result returned to the SQL Server process. The result is then returned to the client.

This process also schedules the population of the full-text indexes and monitors the full-text catalogs.

Replication Component

Replication allows DBAs to copy databases to different locations and keep the copies synchronized. This can be used for data distribution, synchronization, fault tolerance, disaster recovery, load balancing, or testing. The Replication component manages database replication and interacts primarily with the Database Engine component, as shown in Figure 15.9.

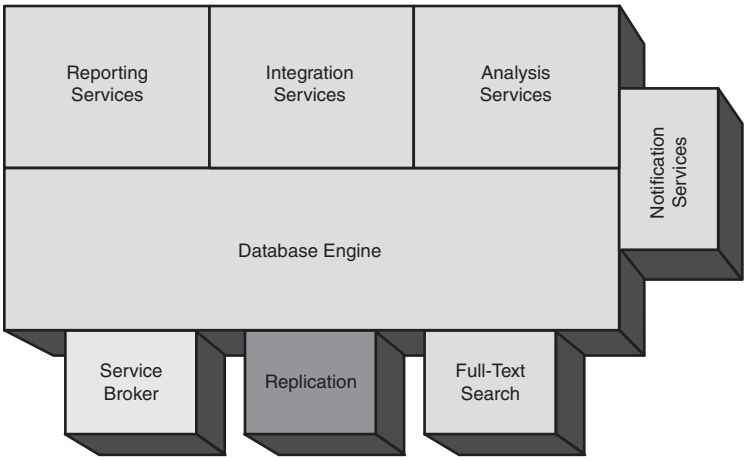

FIGURE 15.9
Replication component.

It is important to understand the underlying model and the components of replication to administer it properly.

Publishing Model

The Replication component uses a publishing model to replicate databases and objects. The model allows for bidirectional replication, although unidirectional replication is more the norm.

The roles in the publishing model are

- **Publisher**—A publisher is a database instance that replicates some data. A publisher can publish more than one publication.
- **Publication**—A publication is a logically defined set of data to be replicated. It contains one or more articles.
- **Article**—An article is a database object that is replicated. It can be a table, view, stored procedure, or other database object.
- **Distributor**—A distributor is a database instance that stores the status and information about the publications. Each publisher is directly associated with a distributor, which can be the same database instance as the publisher (local distributor) or a separate database instance (remote distributor). Remote distributors can support multiple publishers.
- **Subscriber**—A subscriber is a database instance that receives the replicated data. Each subscriber can subscribe to multiple publications from multiple publishers. Subscriptions can be two-way, meaning changes can be replicated back to the publisher in some cases.
- **Subscriptions**—A subscription defines what publications are replicated and on what schedule. It can be either a push or pull subscription.

Together, these roles comprise the replication topology, which is the relationship between the roles and how data flows between them.

Types of Replication

The replication component can use three types of replication:

- **Transactional Replication**—This type of replication takes an initial snapshot and then replicates the transactions as they occur. This allows the replication to take place almost in real-time and to support high volumes of changes. Transactional replication also supports a peer-to-peer mode, which has two instances operating as both publisher and subscriber.
- **Merge Replication**—Data is replicated out from the publisher and then back from the subscribers. This type of replication allows changes to be made at both the publisher and subscriber and then merges the

data in a batch process. This allows for controlled bidirectional updates, but also has mechanisms for handling conflicts in the updates.

- **Snapshot Replication**—Data is replicated out at once in a copy. This form of replication is best suited where data changes infrequently or where the subscriber does not need to be updated in real-time. Snapshot replication is also used to start the other forms of replication.

The specific type of replication chosen depends on the application needs. See the chapters in Part V, "Disaster Recovery and High Availability," for more details on the options for replication.

Replication Agents

SQL Server 2005 replication uses a number of agents to effect replication. These agents are programs run as SQL Server jobs by the SQL Server agent.

The agents are

- **Snapshot Agent**—This agent is used by all three types of replication. It sets up the initial data files and updates the synchronization information in the distribution database. It runs on the distributor role.
- **Log Reader Agent**—This agent is used by transactional replication. There is an instance of this agent on the distributor role for each database using transactional replication. The agent moves the transactions from the publisher to the distributor database.
- **Distribution Agent**—This agent is used by both transactional replication and snapshot replication. It creates the initial snapshot on the subscriber and moves transactions to the subscriber from the distribution database. This agent runs on the distributor for push replication or on the subscriber for pull replication.
- **Merge Agent**—This agent works with merge replication, creating the initial snapshot on the subscribers. It then transfers and reconciles data changes. Each merge subscription has its own instance of this agent. This agent runs on the distributor for push replication or on the subscriber for pull replication.
- **Queue Reader Agent**—This agent works with transactional replication and runs on the distributor. This agent is used only when the queued updating option is used, which allows changes made at the subscriber to be replicated back to the publisher.

These agents handle the actual work of the replication process and run on a scheduled basis.

Service Broker Component

The Service Broker component provides support for loosely coupled applications that need to communicate asynchronously. The service broker allows messages sent between applications with guaranteed delivery. This can be used to provide reliable queries and data collection, distributed applications that access multiple SQL Server 2005 instances, and large-scale batch processing.

Like the replication and full-text search components, the Service Broker component interacts with the Database Engine component, shown in Figure 15.10. The architecture of the Service Broker component consists of conversations, services, and security.

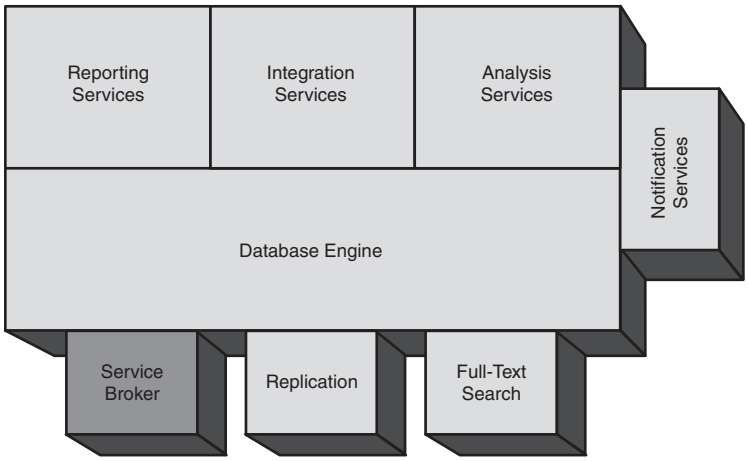

FIGURE 15.10
Service Broker component.

Conversations

The Service Broker component uses a conversation model for communication between applications. These conversations are architected to be persistent, reliable, and asynchronous. The conversations use messages to exchange information. The conversations between two services are known as *dialog conversations*. The dialog conversations provide exactly-once-in-order (EOIO) message delivery, which ensures that messages are delivered exactly once and in the correct order.

The conversations are organized into conversation groups to allow for related conversations to coordinate messages without having the member services directly coordinate with each other.

Conversations are specified by contracts, which describe the direction and types of messages that will be delivered in the dialog.

Each conversation is secured with certificates that provide for end-to-end encryption and remote authorization.

Services

Service broker services are used to describe business tasks. Conversations are established between the services, and each service has its own queue. The service name is used by the service broker to deliver messages to the correct service queue.

SQL Server 2005 Editions

SQL Server 2005 comes in a variety of editions that are tailored to suit the needs and requirements of different organizations and applications.

Enterprise

The SQL Server 2005 Enterprise Edition is the complete feature set of the product and is designed to support the needs of the largest enterprises. It includes features for high scalability and availability, as well as features to support business intelligence applications. The Enterprise Edition is fully 64-bit capable and is optimized to run on 64-bit platforms.

Standard

The SQL Server 2005 Standard Edition includes the core set of functionality needed to support data warehouses, electronic commerce applications, and line-of-business applications. It is designed to support the needs of small to medium organizations. The Standard Edition is fully 64-bit capable.

Workgroup

The SQL Server 2005 Workgroup Edition is designed for small organizations and includes the core database features needed for applications.

Express

The SQL Server 2005 Express Edition is the free edition that is designed to support small or targeted applications with a core set of secure database requirements. This edition replaces the MSDE platform.

Compact

The SQL Server 2005 Compact Edition is the free edition that runs on mobile devices as well as desktops. This provides a single lightweight database platform for client applications. This edition replaces the SQL Server Mobile product.

Developer

The SQL Server 2005 Developer Edition provides all the same features and functionality as the Enterprise Edition but is licensed only for development purposes.

Summary

With a sophisticated and multifaceted platform such as SQL Server 2005, there is an inherent degree of complexity. As this chapter has shown, this complexity can be understood and managed if taken one component at a time.

This chapter conducted a high-level walkthrough of the SQL Server 2005 features and components, discussing the basic terminology and architectures of the components. This information gives you a foundation and a starting point from which to launch into the balance of the book.